

## Exercise 1 – Tapio Koskinen

Screen captures of the code Exercise1.py:

part 1 code and output

```
#Object oriented programming
#Exercise 1

#Part 1.

print("Hello world!")

\ex1.py
Hello world!
>>>
```

part 2

```
#Part 2

#Create two empty lists for strings and numbers

number_list = [0 for i in range(10)]
string_list = ["" for i in range(10)]

#First we ask the user to give numbers for the "empty" slots with a for loop
for i in range(len(number_list)):
    handler = int(input("Give me a number for the "+str(i+1)+" item in number list: "))
    number_list[i] = handler

#Next we do the same for the string list
for i in range(len(string_list)):
    handler = input("Give me a string for the "+str(i+1)+" item in string list: ")
    string_list[i] = handler

#Print out the new lists

print("Your numberlist:",number_list,"\nYour stringlist:",string_list)

#Replace the numberlist with randomly generated numbers

for i in range(len(number_list)):
    number_list[i] = random.randint(0,99)

#Print out the new numberlist with random numbers
print("Numberlist with random numbers:",number_list)
```

### part 3 code and output

```
#Part 3
```

```
#The simplest way to do this is to use python's own sort() function. To ignore  
#cases in stringlist I used sorted(list, key) method.
```

```
number_list.sort()  
string_list = sorted(string_list, key=str.casefold)  
  
print("Sorted numberlist in ascending order:", number_list)  
print("Sorted stringlist in ascending order", string_list)
```

line 56, Column 1

```
Give me a string for the 1 item in string list: Hello  
Give me a string for the 2 item in string list: There  
Give me a string for the 3 item in string list: my  
Give me a string for the 4 item in string list: animal  
Give me a string for the 5 item in string list: friend  
Give me a string for the 6 item in string list: Are  
Give me a string for the 7 item in string list: you  
Give me a string for the 8 item in string list: there  
Give me a string for the 9 item in string list: Or  
Give me a string for the 10 item in string list: not  
Numberlist with random numbers: [78, 44, 78, 74, 79, 98, 28, 1, 55, 14]  
Sorted numberlist in ascending order: [1, 14, 28, 44, 55, 74, 78, 78, 79, 98]  
Sorted stringlist in ascending order ['animal', 'Are', 'friend', 'Hello', 'my', 'not',  
'Or', 'There', 'there', 'you']  
>>>
```

#### Part 4 code and output

```
#Part 4

def negatives(your_numbers):
    negatives = 0
    for i in your_numbers:
        if i < 0:
            negatives += 1
        else:
            continue

    return negatives

def main():

    integer_list = []
    handler = 0
    asking = True
    while asking:
        try:
            handler = int(input("Give me any number: "))
            if handler != 0:
                integer_list.append(handler)
            else:
                asking = False
        except ValueError:
            print("Only numbers please")

    print("The number of negative numbers in list is:",negatives(integer_list))

main()
```

26, Column 41

```
Give me any number: t
Only numbers please
Give me any number: ?
Only numbers please
Give me any number: fkp0
Only numbers please
Give me any number: 3
Give me any number: 1
Give me any number: #
Only numbers please
Give me any number: 5
Give me any number: -2
Give me any number: -6
Give me any number: 1
Give me any number: 16
Give me any number: -22
Give me any number: 0
The number of negative numbers in list is: 3
```

## Part 5 code and output

```
#Part 5

def even_numbers(your_numbers):
    even_numbers = 0
    for i in your_numbers:
        if i % 2 == 0:
            even_numbers += 1
        else:
            continue
    return even_numbers

def main():

    integer_list = []
    handler = 0
    asking = True
    while asking:
        try:
            handler = int(input("Give me any number: "))
            if handler != 0:
                integer_list.append(handler)
            else:
                asking = False
        except ValueError:
            print("Only numbers please")

    #print("The number of negative numbers in list is:",negatives(integer_list))

    print("The number of even numbers in list is:",even_numbers(integer_list))

main()
```

File 41, Column 79

```
Give me any number: 5
Give me any number: 14
Give me any number: 10
Give me any number: 22
Give me any number: 43
Give me any number: -12
Give me any number: -15
Give me any number: -4
Give me any number: 0
The number of even numbers in list is: 5
>>>
```

## Part 6 code and output

```
#Part 6

def sum_of_div_3(your_numbers):
    positive_div_threes = []
    for i in your_numbers:
        if i > 0:
            if i % 3 == 0:
                positive_div_threes.append(i)
            else:
                continue
        else:
            continue
    the_sum = sum(positive_div_threes)
    return the_sum

def main():

    integer_list = []
    handler = 0
    asking = True
    while asking:
        try:
            handler = int(input("Give me any number: "))
            if handler != 0:
                integer_list.append(handler)
            else:
                asking = False
        except ValueError:
            print("Only numbers please")

    #print("The number of negative numbers in list is:", negatives(integer_list))

    #print("The number of even numbers in list is:", even_numbers(integer_list))

    print("The sum of positive numbers divisible by 3 is", sum_of_div_3(integer_list))

main()
```

Ln 58, Column 85

```
>>>
Give me any number: 3
Give me any number: 6
Give me any number: 9
Give me any number: 5
Give me any number: 10
Give me any number: 15
Give me any number: -3
Give me any number: -6
Give me any number: -9
Give me any number: 0
The sum of positive numbers divisible by 3 is 33
>>>
```

Ln: 431 Col: 4



## Part 7 code and output

```
#Part 7

def ap_counter(your_number):
    ap_numbers = []
    your_number -= your_number % 2

    for i in range(2, your_number+2, 2):
        ap_numbers.append(i)

    return ap_numbers

def sum_of_ap(your_ap):

    return sum(your_ap)

def sum_of_squared_ap(your_ap):

    the_squared_sum = 0
    for i in your_ap:
        the_squared_sum += i**2

    return the_squared_sum

def main():

    asking = True
    while asking:
        try:
            user_input = int(input("Slap a number on me: "))
            asking = False
        except ValueError:
            print("Only number please")

    ap_list = ap_counter(user_input)

    print("The arithmetic progression(2) from your number is:",ap_list)
    print("The sum of those numbers is",sum_of_ap(ap_list))
    print("The sum of squared terms in progression is",sum_of_squared_ap(ap_list))

main()
```

41, Column 7

```
Slap a number on me: 10
The arithmetic progression(2) from your number is: [2, 4, 6, 8, 10]
The sum of those numbers is 30
The sum of squared terms in progression is 220
>>>
```

Ln: 439 Col:

## Part8

```
#Part 8

import random

#choose number function works as the input handler

def choose_number():
    choosing = True
    while choosing:
        try:
            user_choise = int(input("Choose by typing the number Rock(1), Paper(2) or Scissors(3): "))
            if user_choise < 1 or user_choise > 3:
                print("Choose the number 1, 2 or 3")
            else:
                choosing = False
        except ValueError:
            print("Choose the number 1, 2 or 3")
    return user_choise

#Check score function check if there is a winner

def check_score(score):
    for key in score:
        if score[key] >= 3:
            return key

#Play again asks if the player wants to play again and the output is a boolean for the game loop

def play_again():
    global Score
    choosing = True
    while choosing:
        try:
            play_again = int(input("Play again? yes=1 no=2: "))
            if play_again == 1:
                Score = {"Player":0,"Machine":0}
                return True
            elif play_again == 2:
                return False
            else:
                print("1 or 2")
        except ValueError:
            print("1 or 2")
```

```

#Main program

def main():

    the_game = True

    while the_game:

        print("Lets play rock paper scissors! Best out of three!")

        playing = True

        weapons = ["filler", "Rock", "Paper", "Scissors"]

        Score = {"Player":0,"Machine":0}

        #This could be on a function on its own, but I thought it's simpler to just have it as such
        while playing:

            player = choose_number()

            machine = random.randint(1,3)

            if player == machine:
                print("Tie!")

            elif player == 1:
                if machine == 2:
                    print("You lose.",weapons[machine],"beats",weapons[player],".")
                    Score["Machine"] += 1
                elif machine == 3:
                    print("You win!",weapons[player],"beats",weapons[machine],".")
                    Score["Player"] += 1
            elif player == 2:
                if machine == 1:
                    print("You win!",weapons[player],"beats",weapons[machine],".")
                    Score["Player"] += 1
                elif machine == 3:
                    print("You lose.",weapons[machine]," beats",weapons[player],".")
                    Score["Machine"] += 1
            elif player == 3:
                if machine == 1:
                    print("You lose.",weapons[machine],"beats",weapons[player],".")
                    Score["Machine"] += 1
                elif machine == 2:
                    print("You win!",weapons[player],"beats",weapons[machine],".")
                    Score["Player"] += 1

            if check_score(Score) == "Machine":
                print("Machine won 3 times. You lose.")
                playing = False
            elif check_score(Score) == "Player":
                print("You won 3 times! You win the game!")
                playing = False

        the_game = play_again()

    print("Thanks for playing!")

main()

```



Part9

```
#Part 9

import random

def random_number():

    return random.randint(1,6)

print("A random number between 1 and 6:",random_number())

|
```

e 11, Column 1

Tab Si

```
ObjectOrientedProgramming\Exercisel\Ex1(4).py
A random number between 1 and 6: 3
>>>
```

Ln: 24 Col: 4

Screen capture of the output:

#### Part 2 output

```
Give me a number for the 1 item in number list: 9
Give me a number for the 2 item in number list: 2
Give me a number for the 3 item in number list: 7
Give me a number for the 4 item in number list: 4
Give me a number for the 5 item in number list: 3
Give me a number for the 6 item in number list: 9
Give me a number for the 7 item in number list: 11
Give me a number for the 8 item in number list: 51
Give me a number for the 9 item in number list: 2
Give me a number for the 10 item in number list: 4
Give me a string for the 1 item in string list: Get
Give me a string for the 2 item in string list: ready
Give me a string for the 3 item in string list: for
Give me a string for the 4 item in string list: the
Give me a string for the 5 item in string list: ultimate
Give me a string for the 6 item in string list: string
Give me a string for the 7 item in string list: list
Give me a string for the 8 item in string list: tournament
Give me a string for the 9 item in string list: fighter
Give me a string for the 10 item in string list: game
Your numberlist: [9, 2, 7, 4, 3, 9, 11, 51, 2, 4]
Your stringlist: ['Get', 'ready', 'for', 'the', 'ultimate', 'string', 'list', 'tournament', 'fighter', 'game']
Numberlist with random numbers: [94, 87, 18, 16, 97, 95, 30, 52, 99, 45]
>>>
```

## Part 8 output

```
Lets play rock paper scissors! Best out of three!
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 1
Tie!
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 3
You win! Scissors beats Paper .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 2
You win! Paper beats Rock .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 1
You lose. Paper beats Rock .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 3
You lose. Rock beats Scissors .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 2
You win! Paper beats Rock .
You won 3 times! You win the game!
Play again? yes=1 no=2: 1
Lets play rock paper scissors! Best out of three!
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 2
You lose. Scissors beats Paper .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 1
You win! Rock beats Scissors .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 3
You lose. Rock beats Scissors .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 1
You win! Rock beats Scissors .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 2
You win! Paper beats Rock .
You won 3 times! You win the game!
Play again? yes=1 no=2: 1
Lets play rock paper scissors! Best out of three!
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 2
You win! Paper beats Rock .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 3
Tie!
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 1
Tie!
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 2
Tie!
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 1
Tie!
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 2
You lose. Scissors beats Paper .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 2
You lose. Scissors beats Paper .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 3
You win! Scissors beats Paper .
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 2
Tie!
Choose by typing the number Rock(1), Paper(2) or Scissors(3): 1
You win! Rock beats Scissors .
You won 3 times! You win the game!
Play again? yes=1 no=2: 2
Thanks for playing!
>>> |
```

**Part 10** Explain the following terms (use your own words, do not copy paste answers from Internet).  
You can answer in Finnish or English.

**a. Procedural programming**

Procedural programming is a structure where the code is divided into independent subprograms or procedures. In other words it goes through simple instructions one by one until the end. The idea is to make operations on data.

**b. Functional programming**

Functional programming is about passing data from function to function, so compared to procedural, it does not necessarily need to go one by one from top to down.

**c. Object oriented programming**

OOP is a combination of the two previous ways, but the main difference is the implication of objects and classes. These in combination help construct a simpler and more efficient program. "Don't repeat yourself"-principle applies here well, because you can reuse these objects.

**d. Class (in programming)**

Classes are used to group objects together. For example a class could be a car-brand and the objects of this class would be volvo, bmw, audi etc.

**e. Object (in programming)**

Objects contain data, from simple to functional. It is always under a class.

**f. Instance (in programming)**

An instance refers to a situation where an object is called, there it is called an instance of an object.

**g. Encapsulation (in programming)**

When data and methods are bundled, it considered as an encapsulation. There after the encapsulation can be considered as a class, containing objects.

### Screen capture of the Git status:

The screenshot shows the GitHub interface for the repository 'tadvpio / Object-Oriented-Programming'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, there are buttons for 'main' branch, '1 branch', and '0 tags'. To the right are buttons for 'Go to file', 'Add file', and 'Code'. The main content area displays a commit history table for the user 'tadvpio'.

Commit Hash	Commit Message	Time
3114627	Add files via upload	32 minutes ago
	Ex1(1).py	32 minutes ago
	Ex1(2).py	32 minutes ago
	Ex1(3).py	32 minutes ago
	Ex1(4).py	32 minutes ago
	ex1.py	32 minutes ago

### Self-assessment:

I feel like I have improved quite a bit in a year with python and programming in general. This makes me feel very happy. It took me about an hour to write the programming exercises and another hour to do the rest. I'm still a bit uncertain of the constructing ideology in OOP(even if it is fundamental), but I feel like later in this course I will understand way better than now.

I bundled the code and output to a same picture to save space and time, apart from part 2 and 8.

I added the files to git before making a subfolder for exercise\_1, therefore they are at the top as separate files. Later I will add folders per exercise.

Looking forward to learn more!